A HYBRID PARALLEL SOLVER FRAMEWORK FOR GENERAL SPARSE LINEAR SYSTEMS

SIVASANKARAN RAJAMANICKAM, ERIK G. BOMAN, AND MICHAEL A. HEROUX

1. Introduction

As parallelism in a single node increases, computational science applications have to adapt to a hybrid system where each compute node by itself is a shared memory system. This presents new challenges and opportunities for robust sparse linear solvers and preconditioners on the node. For example, domain decomposition based preconditioners can scale better with node level sparse linear solver/preconditioner as the number of subdomains can be limited to the number of nodes (leading to fewer iterations). Even within a node, high performance solvers should account for NUMA based architectures. We present a hybrid solver, HyperLU, for solving general sparse linear systems on the node. HyperLU can also be used as a preconditioner.

2. Solver Framework

Our approach includes two levels of parallelism, where the top level is based on exploiting bordered block structure. Although such structure sometimes occurs naturally, we use hypergraph partitioning to find such block structure. Solvers that use this idea have been implemented in distributed memory architectures [1]. We use a similar framework for our solver, but limit the subdomains (and MPI tasks) to the NUMA regions in a shared memory node. Each subdomain (diagonal block) can be solved either exactly or ineaxactly. Note that any multithreaded sparse solver can be used here to exploit fine-grain parallelism.

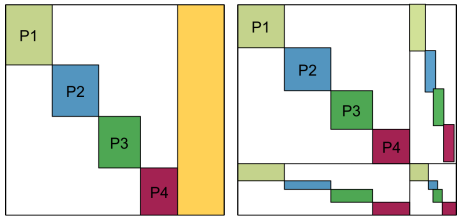

Figure 1. Column ordering and symmetric reordering using hypergraph partitioning.

Figure 1(a) shows a matrix A with column ordering from a hypergraph partition. The hypergraph model in general can handle unsymmetric sparse matrices. It is also more suitable for partial pivoting on the columns and for a left-looking sparse LU factorization on each of the subdomain (as no row is shared between two subdomains). For symmetric matrices, the column ordering can also be used as a row ordering to get square subdomains, and the reordered matrix will have the structure shown in figure 1(b). Once the subdomains are solved in parallel we can solve for the Schur complement to solve for A. The Schur complement is never explicitly computed. We compute an approximate Schur complement using probing for a predetermined structure and use an inner iteration to solve it. This results in a much smaller system for the iterative methods. We can also replace the solve on the subdomains with incomplete factorization algorithms and use this framework as a node level hybrid preconditioner in a global solve.

3. Experimental Results

HyperLU is currently implemented as a preconditioner in the Trilinos framework [2] with AztecOO as the iterative solver. The serial solver KLU is used to solve each subdomain, as our multithreaded version is still under development. All the experiments use Zoltan as the hypergraph partitioner to partition for four subdomains (for the NUMA

Scalable Algorithms Department, Sandia National Laboratories, Albuquerque, NM. Sandia is a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

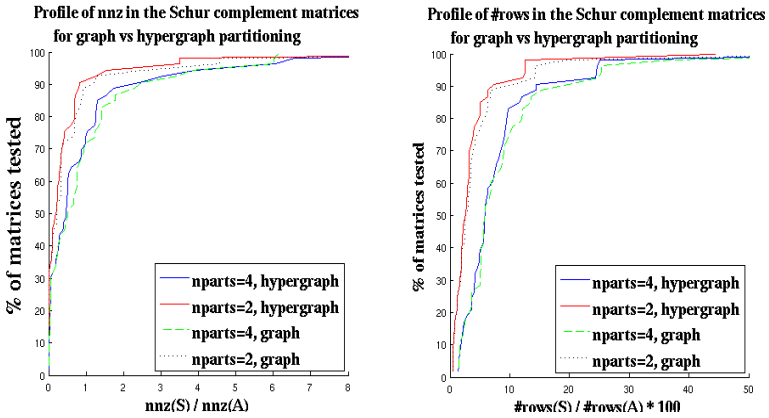

FIGURE 2. Comparison of the size of the Schur complement for graph and hypergraph partitioning.

regions/sockets). The tests use symmetric matrices of the size 10K to 20K rows from the UF sparse matrix collection, one matrix from a Sandia application (Tramanto) and a synthetic FEM matrix from MATLAB (wathenLarge).

Trilinos framework has no support for hybrid methods like HyperLU where the iterative method can be used just on the Schur complement. As a result, the experiments iterate over the entire matrix, even when the subdomains were solved exactly using KLU. The results will get better as the software can be adapted to iterate on the Schur complement. Table 1 shows the number of iterations for AztecOO to converge for HyperLU, ML (Algebraic multilevel method) and two incomplete factorizations. A '-' in the number of iterations show that the method did not converge for that test case. HyperLU is as good as or better than the other incomplete factorizations for these tests. The parameters of ILU and ILUT can be modified for better performance than shown here. The experiments shown here uses the default values level of fill zero for ILU and ILUT with level of fill as one and drop tolerance as $1e-12$.

Matrix Name	HyperLU	ML	ILU	ILUT
Cage11	13	14	12	12
cbuckle	101	-	-	-
Lourakis	28	20	42	38
FIDAPex35	16	-	-	-
Oberwolfach	-	27	-	-
fem_3d_thermal	25	23	26	25
Dubkova1	56	55	189	154
Tramanto	112	-	-	-
wathenLarge	35	14	36	37

4. OBSERVATIONS

Although the hypergraph based nested dissection exposes parallelism for our solver framework, the objective for the partitioning in our problem is different than the objective for hypergraph partitioning. The ideal partitioning objective for HyperLU style solver is that subdomains be better balanced, with a small separator and a sparser (or better structured) Schur complement. No partitioners solve this objective directly. However, graph partitioning can be used instead of hypergraph partitioning as the test matrices were symmetric. Figure 2 compares graph (METIS) and hypergraph (Patoh) partitioning with 53 matrices of size 1K to 10K rows from the UF sparse matrix collection. We explicitly compute the Schur complement in MATLAB for these matrices. The improvement in hypergraph partitioner over graph partitioner in terms of the number of non-zeros in the Schur complement is marginal for this set of matrices. However, hypergraph partitioning is best suited for our approach of using a left-looking solver that uses partial pivoting in the subdomains.

REFERENCES

[1] I. S. Duff and J. A. Scott. A parallel direct solver for large sparse highly unsymmetric linear systems. *ACM Trans. Math. Softw.*, 30(2):95–117, 2004.

[2] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.